PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2003-330992
(43)Date of publication of application : 21.11.2003

(51)Int.Cl. G06F 17/50

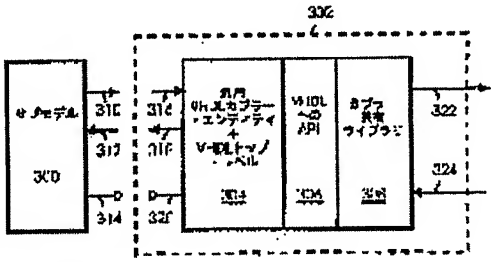
(21)Application number : 2003-100239
(22)Date of filing : 03.04.2003

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : MELLORS WILLIAM K
RICH MARVIN J

(30)Priority
Priority number : 2002 125198 Priority date : 18.04.2002 Priority country : US

(54) MODEL SIMULATION

(57)Abstract:
PROBLEM TO BE SOLVED: To provide techniques for tightly functionally coupling submodels of a partitioned model for concurrent distributed submodel simulation runs.
SOLUTION: A technique for distributed processing a partitioned model is provided based on tight functional coupling of multiple submodels of the model. The technique includes, in one embodiment, providing each submodel 300 with a generic coupler 302 to enable processing of the submodel 300 on any simulator instance of any simulator. Submodels 300 coupled with the generic couplers 302 can be processed on the same or different coupling units. The generic couplers 302 facilitate communication between submodels 300 through a common communication directory (CCD) by using functions of a generic coupler shared library 308. The generic couplers 302 further use functions of the shared library 308 to ensure integrity of data transmitted between submodels 300.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2003－330992
（P2003－330992A）
(43)公開日　平成15年11月21日(2003.11.21)

(51)Int.Cl.⁷	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　17/50	６６４	Ｇ０６Ｆ　17/50	６６４Ｌ　５Ｂ０４６

審査請求　有　請求項の数48　ＯＬ　（全 19 頁）

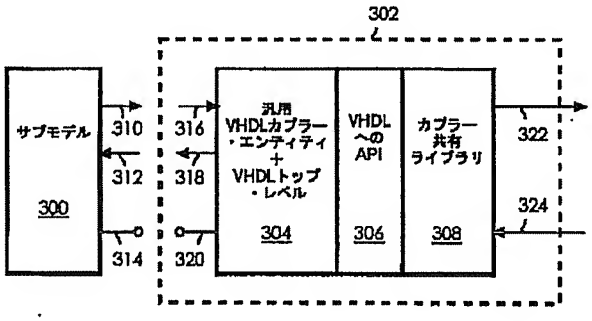
(21)出願番号　特願2003－100239(P2003－100239)
(22)出願日　平成15年４月３日(2003.4.3)
(31)優先権主張番号　１０／１２５１９８
(32)優先日　平成14年４月18日(2002.4.18)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100086243
弁理士　坂口　博　（外２名）

最終頁に続く

(54)【発明の名称】　モデル・シミュレーション

(57)【要約】
【課題】　分割されたモデルのサブモデル群をコンカレント分散サブモデル・シミュレーション実行できるように緊密かつ機能的に結合する手法を提供する。
【解決手段】　分割されたモデルの複数のサブモデルを緊密かつ機能的に結合することにより分割されたモデルを分散処理する手法を提供する。この手法は、一例では、任意のシミュレータの任意のシミュレータ・インスタンス上でサブモデル３００の処理を可能にする汎用カプラー３０２を各サブモデル３００に備えるステップを備えている。汎用カプラー３０２に結合されたサブモデル３００は同じまたは異なるコンピューティング・ユニットで処理することができる。汎用カプラー３０２は汎用カプラー共有ライブラリ３０８の機能を使うことにより、共通通信ディレクトリ（ＣＣＤ）を通じたサブモデル３００間の通信を容易にする。汎用カプラー３０２はさらに、汎用カプラー共有ライブラリ３０８の機能を使ってサブモデル３００間で転送されるデータの一貫性を保証する。



1

【特許請求の範囲】

【請求項１】分割されたモデルの、第１のサブモデルおよび第２のサブモデルを含む複数のサブモデルを処理する方法であって、

分割されたモデルの第１のサブモデルと第２のサブモデルとを結合するステップであって、

前記第１のサブモデルがインタフェースする第１のカプラーを準備するステップと、

前記第２のサブモデルがインタフェースする第２のカプラーを準備するステップと、

前記第１のカプラーと前記第２のカプラーとをインタフェースさせるステップであって、前記第１のカプラーと前記第２のカプラーとの間にバッファを準備するステップとを備えたステップと、

前記第１のサブモデルおよび前記第２のサブモデルを一括して処理するステップであって、前記第１のサブモデルおよび前記第２のサブモデルが前記第１のカプラー、前記バッファ、および前記第２のカプラーを用いて通信する、ステップとを備えたステップを備えた方法。

【請求項２】前記モデルが論理設計である、請求項１に記載の方法。

【請求項３】前記一括して処理するステップが前記第１のサブモデルおよび前記第２のサブモデルを分散処理するステップを備え、

前記分散処理するステップが分散プロセス制御を備えている、請求項１に記載の方法。

【請求項４】前記一括して処理するステップがさらに前記第１のサブモデルを第１のシミュレータ・インスタンス上で処理するステップと前記第２のサブモデルを第２のシミュレータ・インスタンス上で処理するステップとを備え、

前記第１のシミュレータ・インスタンスは前記第２のシミュレータ・インスタンスと異なり、

前記第１のシミュレータ・インスタンスは前記バッファを用いて前記第２のシミュレータ・インスタンスと通信する、請求項１に記載の方法。

【請求項５】前記第１のシミュレータ・インスタンスは第１のコンピューティング・ユニットに常駐し、前記２のシミュレータ・インスタンスは第２のコンピューティング・ユニットに常駐する、請求項４に記載の方法。

【請求項６】少なくとも１つのサブモデルおよび付随するカプラーが少なくとも１つのマップされたデータ・ポートおよびクロック／サイクル・ポートを備え、

前記クロック／サイクル・ポートは少なくとも１つのクロック／サイクル信号を供給し、

前記少なくとも１つのクロック／サイクル信号は前記少なくとも１つのサブモデルと付随するカプラーとの間で前記少なくとも１つのマップされたデータ・ポートを通じてデータを転送するのに使用される、請求項１に記載の方法。

2

【請求項７】前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方がトップ・レベル・カプラー・エンティティとカプラー共有ライブラリとを備え、

前記第１のカプラーおよび前記第２のカプラーのうちの前記少なくとも一方が前記トップ・レベル・カプラー・エンティティを用いて前記第１のサブモデルおよび前記第２のサブモデルと個別に接続し、

前記トップ・レベル・カプラー・エンティティが少なくとも１つの入力と、少なくとも１つの出力と、前記第１のサブモデルおよび前記第２のサブモデルのうちの少なくとも一方の前記クロック／サイクル・ポートにマップされた複数のポートとを備え、

前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方が前記カプラー共有ライブラリを用いて前記カプラーと前記バッファとの間でデータを転送する、請求項６に記載の方法。

【請求項８】前記バッファが通信媒体を備え、

前記通信媒体は前記第１のカプラーと前記第２のカプラーとの間でデータを渡すとともに監視する少なくとも１つのプロトコルを備えている、請求項１に記載の方法。

【請求項９】前記一括して処理するステップがさらに、

前記第１のカプラーが前記第１のサブモデルから取得したデータを前記バッファに送信し、前記第２のカプラーが前記データを前記バッファから取得し、前記データを前記第２のサブモデルに供給するステップと、

前記送信と前記取得とを対応させることにより、前記データの一貫性を保証するステップとを備えた、請求項１に記載の方法。

【請求項１０】前記一貫性を保証するステップが、

前記データにチェックサム値を挿入するステップと、

前記第１のカプラーおよび前記第２のカプラーのうちの一方が前記バッファ中の前記データを調べて前記データの正確性を判断するステップと、

前記データが不正確な場合、前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方が、前記データの前記バッファへの再送信および前記データの前記バッファからの再読み取りのうちの少なくとも一方を行なうステップとを備えた、請求項９に記載の方法。

【請求項１１】前記挿入するステップが、

前記チェックサム値を計算するステップを備え、

前記計算するステップが、

少なくとも１つの論理操作を前記データに適用するステップを備えている、請求項１０に記載の方法。

【請求項１２】前記調べるステップが、

前記データの新たなチェックサム値を計算するステップと、

前記新たなチェックサム値と前記チェックサム値とを比較するステップと備えた、請求項１０に記載の方法。

【請求項１３】前記一貫性を保証するステップがさらに、

3

前記バッファに、前記第１のカプラーが書き込むステップと

4

前記バッファに、前記第１のカプラーが書き込みアクセス可能であるとともに前記第２のカプラーが読み取りアクセス可能な第１の一連ファイルを準備するステップと、

前記バッファに、前記第１のカプラーが読み取りアクセス可能であるとともに前記第２のカプラーが書き込みアクセス可能な第２の一連ファイルを準備するステップと、

所定の順序で、前記第１のカプラーが前記第１の一連ファイルに書き込み、前記第２のカプラーが前記第１の一連ファイルから読み取るステップと、

前記所定の順序で、前記第２のカプラーが前記第２の一連ファイルに書き込み、前記第１のカプラーが前記第２の一連ファイルから読み取るステップとを備えた、請求項９に記載の方法。

【請求項１４】前記インタフェースさせるステップがさらに、前記バッファ中のデータを監視して監視統計を供給するステップを備え、

前記監視統計が前記第１のサブモデルおよび前記第２のサブモデルを統計的に再分割して前記一括処理を最適化するのを容易にする、請求項１に記載の方法。

【請求項１５】前記モデルがＶＨＤＬで表されている、請求項１に記載の方法。

【請求項１６】分割されたモデルの、第１のサブモデルおよび第２のサブモデルを含む複数のサブモデルを処理するシステムであって、

分割されたモデルの第１のサブモデルと第２のサブモデルとを結合する手段であって、

前記第１のサブモデルがインタフェースする第１のカプラーを準備する手段と、

前記第２のサブモデルがインタフェースする第２のカプラーを準備する手段と、

前記第１のカプラーと前記第２のカプラーとをインタフェースさせる手段であって、前記第１のカプラーと前記第２のカプラーとの間にバッファを準備する手段とを備えた手段と、

前記第１のサブモデルおよび前記第２のサブモデルを一括して処理する手段であって、前記第１のサブモデルおよび前記第２のサブモデルが前記第１のカプラー、前記バッファ、および前記第２のカプラーを用いて通信する、手段とを備えた手段を備えた方法。

【請求項１７】前記モデルが論理設計である、請求項１６に記載のシステム。

【請求項１８】前記一括して処理する手段が前記第１のサブモデルおよび前記第２のサブモデルを分散処理する手段を備え、

前記分散処理する手段が分散プロセス制御を備えている、請求項１６に記載のシステム。

【請求項１９】前記一括して処理する手段がさらに前記第１のサブモデルを第１のシミュレータ・インスタンス上で処理する手段と前記第２のサブモデルを第２のシミュレータ・インスタンス上で処理する手段とを備え、

前記第１のシミュレータ・インスタンスは前記第２のシミュレータ・インスタンスと異なり、

前記第１のシミュレータ・インスタンスは前記バッファを用いて前記第２のシミュレータ・インスタンスと通信する、請求項１６に記載のシステム。

【請求項２０】前記第１のシミュレータ・インスタンスは第１のコンピューティング・ユニットに常駐し、前記２のシミュレータ・インスタンスは第２のコンピューティング・ユニットに常駐する、請求項１９に記載のシステム。

【請求項２１】少なくとも１つのサブモデルおよび付随するカプラーが少なくとも１つのマップされたデータ・ポートおよびクロック／サイクル・ポートを備え、

前記クロック／サイクル・ポートは少なくとも１つのクロック／サイクル信号を供給し、

前記少なくとも１つのクロック／サイクル信号は前記少なくとも１つのサブモデルと付随するカプラーとの間で前記少なくとも１つのマップされたデータ・ポートを通じてデータを転送するのに使用される、請求項１６に記載のシステム。

【請求項２２】前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方がトップ・レベル・カプラー・エンティティとカプラー共有ライブラリとを備え、

前記第１のカプラーおよび前記第２のカプラーのうちの前記少なくとも一方が前記トップ・レベル・カプラー・エンティティを用いて前記第１のサブモデルおよび前記第２のサブモデルと個別に接続し、

前記トップ・レベル・カプラー・エンティティが少なくとも１つの入力と、少なくとも１つの出力と、前記第１のサブモデルおよび前記第２のサブモデルのうちの少なくとも一方の前記クロック／サイクル・ポートにマップされた複数のポートとを備え、

前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方が前記カプラー共有ライブラリを用いて前記カプラーと前記バッファとの間でデータを転送する、請求項２１に記載のシステム。

【請求項２３】前記バッファが通信媒体を備え、

前記通信媒体は前記第１のカプラーと前記第２のカプラーとの間でデータを渡すとともに監視する少なくとも１つのプロトコルを備えている、請求項１６に記載のシステム。

【請求項２４】前記一括して処理する手段がさらに、

前記第１のカプラーが前記第１のサブモデルから取得したデータを前記バッファに送信し、前記第２のカプラーが前記データを前記バッファから取得し、前記データを前記第２のサブモデルに供給する手段と、

前記送信と前記取得とを対応させることにより、前記デ

ータの一貫性を保証する手段とを備えた、請求項１６に記載のシステム。

【請求項２５】前記一貫性を保証する手段が、
前記データにチェックサム値を挿入する手段と、
前記第１のカプラーおよび前記第２のカプラーのうちの一方が前記バッファ中の前記データを調べて前記データの正確性を判断する手段と、
前記データが不正確な場合、前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方が、前記データの前記バッファへの再送信および前記データの前記バッファからの再読み取りのうちの少なくとも一方を行なう手段とを備えた、請求項２４に記載の方法。

【請求項２６】前記挿入する手段が、
前記チェックサム値を計算する手段を備え、
前記計算する手段が、
少なくとも１つの論理操作を前記データに適用する手段を備えている、請求項２５に記載のシステム。

【請求項２７】前記調べる手段が、
前記データの新たなチェックサム値を計算する手段と、
前記新たなチェックサム値と前記チェックサム値とを比較する手段と備えた、請求項２５に記載のシステム。

【請求項２８】前記一貫性を保証する手段がさらに、
前記バッファに、前記第１のカプラーが書き込みアクセス可能であるとともに前記第２のカプラーが読み取りアクセス可能な第１の一連ファイルを準備する手段と、
前記バッファに、前記第１のカプラーが読み取りアクセス可能であるとともに前記第２のカプラーが書き込みアクセス可能な第２の一連ファイルを準備する手段と、
所定の順序で、前記第１のカプラーが前記第１の一連ファイルに書き込み、前記第２のカプラーが前記第１の一連ファイルから読み取る手段と、
前記所定の順序で、前記第２のカプラーが前記第２の一連ファイルに書き込み、前記第１のカプラーが前記第２の一連ファイルから読み取る手段とを備えた、請求項２４に記載のシステム。

【請求項２９】前記インタフェースさせる手段がさらに、前記バッファ中のデータを監視して監視統計を供給する手段を備え、
前記監視統計が前記第１のサブモデルおよび前記第２のサブモデルを統計的に再分割して前記一括処理を最適化するのを容易にする、請求項１６に記載のシステム。

【請求項３０】前記モデルがＶＨＤＬで表されている、請求項１６に記載のシステム。

【請求項３１】分割されたモデルの、第１のサブモデルおよび第２のサブモデルを含む複数のサブモデルを処理するシミュレーション・システムであって、
分割されたモデルの第１のサブモデルと第２のサブモデルとを、前記第１のサブモデルにインタフェースした第１のカプラーおよび前記第２のサブモデルにインタフェースした第２のカプラーを準備し、前記第１のカプラーと前記第２のカプラーとの間に設けたバッファによって前記第１のカプラーと前記第２のカプラーとをインタフェースさせることにより、結合するように適合した少なくとも１つのコンピューティング・ユニットを備え、
前記少なくとも１つのコンピューティング・ユニットがさらに前記第１のサブモデルおよび前記第２のサブモデルを一括して処理するように適合しており、
前記第１のサブモデルおよび前記第２のサブモデルが前記第１のカプラー、前前記バッファ、および記第２のカプラーを用いて通信するシミュレーション・システム。

【請求項３２】前記少なくとも１つのコンピューティング・ユニットが複数のシミュレータ・インスタンスを備え、
前記複数のシミュレータ・インスタンスが第１のシミュレータ・インスタンスおよび第２のシミュレータ・インスタンスを含み、
前記第１のシミュレータ・インスタンスが前記第１のサブモデルを処理し、前記第２のシミュレータ・インスタンスが前記第２のサブモデルを処理し、
前記第１のシミュレータ・インスタンスが前記第２のシミュレータ・インスタンスとは異なる、請求項３１に記載のシミュレーション・システム。

【請求項３３】前記第１のシミュレータ・インスタンスが第１のコンピューティング・ユニットに常駐し、前記第２のシミュレータ・インスタンスが第２のコンピューティング・ユニットに常駐する、請求項３２に記載のシミュレーション・システム。

【請求項３４】分割されたモデルの、第１のサブモデルおよび第２のサブモデルを含む複数のサブモデルを処理する方法であって、
分割されたモデルの第１のサブモデルと第２のサブモデルとを結合するステップであって、
前記第１のサブモデルがインタフェースする第１のカプラーを準備するステップと、
前記第２のサブモデルがインタフェースする第２のカプラーを準備するステップと、
前記第１のカプラーと前記第２のカプラーとをインタフェースさせるステップであって、前記第１のカプラーと前記第２のカプラーとの間にバッファを準備するステップとを備えたステップと、
前記第１のサブモデルおよび前記第２のサブモデルを一括して処理するステップであって、前記第１のサブモデルおよび前記第２のサブモデルが前記第１のカプラー、前記バッファ、および前記第２のカプラーを用いて通信する、ステップとを備えたステップを備えた方法を実行する、機械によって実行可能な命令から成る少なくとも１つのプログラムを有形的に記録した、機械読み取り可能なプログラム記憶装置。

【請求項３５】前記モデルが論理設計である、請求項３４に記載のプログラム記憶装置。

【請求項３６】前記一括して処理するステップが前記第１のサブモデルおよび前記第２のサブモデルを分散処理するステップを備え、
前記分散処理するステップが分散プロセス制御を備えている、請求項３４に記載のプログラム記憶装置。
【請求項３７】前記一括して処理するステップがさらに前記第１のサブモデルを第１のシミュレータ・インスタンス上で処理するステップと前記第２のサブモデルを第２のシミュレータ・インスタンス上で処理するステップとを備え、
前記第１のシミュレータ・インスタンスは前記第２のシミュレータ・インスタンスと異なり、
前記第１のシミュレータ・インスタンスは前記バッファを用いて前記第２のシミュレータ・インスタンスと通信する、請求項３４に記載のプログラム記憶装置。
【請求項３８】前記第１のシミュレータ・インスタンスは第１のコンピューティング・ユニットに常駐し、前記２のシミュレータ・インスタンスは第２のコンピューティング・ユニットに常駐する、請求項３７に記載のプログラム記憶装置。
【請求項３９】少なくとも１つのサブモデルおよび付随するカプラーが少なくとも１つのマップされたデータ・ポートおよびクロック／サイクル・ポートを備え、
前記クロック／サイクル・ポートは少なくとも１つのクロック／サイクル信号を供給し、
前記少なくとも１つのクロック／サイクル信号は前記少なくとも１つのサブモデルと付随するカプラーとの間で前記少なくとも１つのマップされたデータ・ポートを通じてデータを転送するのに使用される、請求項３４に記載のプログラム記憶装置。
【請求項４０】前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方がトップ・レベル・カプラー・エンティティとカプラー共有ライブラリとを備え、
前記第１のカプラーおよび前記第２のカプラーのうちの前記少なくとも一方が前記トップ・レベル・カプラー・エンティティを用いて前記第１のサブモデルおよび前記第２のサブモデルと個別に接続し、
前記トップ・レベル・カプラー・エンティティが少なくとも１つの入力と、少なくとも１つの出力と、前記第１のサブモデルおよび前記第２のサブモデルのうちの少なくとも一方の前記クロック／サイクル・ポートにマップされた複数のポートとを備え、
前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方が前記カプラー共有ライブラリを用いて前記カプラーと前記バッファとの間でデータを転送する、請求項３９に記載のプログラム記憶装置。
【請求項４１】前記バッファが通信媒体を備え、
前記通信媒体は前記第１のカプラーと前記第２のカプラーとの間でデータを渡すとともに監視する少なくとも１つのプロトコルを備えている、請求項３４に記載のプログラム記憶装置。
【請求項４２】前記一括して処理するステップがさらに、
前記第１のカプラーが前記第１のサブモデルから取得したデータを前記バッファに送信し、前記第２のカプラーが前記データを前記バッファから取得し、前記データを前記第２のサブモデルに供給するステップと、
前記送信と前記取得とを対応させることにより、前記データの一貫性を保証するステップとを備えた、請求項３４に記載のプログラム記憶装置。
【請求項４３】前記一貫性を保証するステップが、
前記データにチェックサム値を挿入するステップと、
前記第１のカプラーおよび前記第２のカプラーのうちの一方が前記バッファ中の前記データを調べて前記データの正確性を判断するステップと、
前記データが不正確な場合、前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方が、前記データの前記バッファへの再送信および前記データの前記バッファからの再読み取りのうちの少なくとも一方を行なうステップとを備えた、請求項４２に記載のプログラム記憶装置。
【請求項４４】前記挿入するステップが、
前記チェックサム値を計算するステップを備え、
前記計算するステップが、
少なくとも１つの論理操作を前記データに適用するステップを備えている、請求項４３に記載のプログラム記憶装置。
【請求項４５】前記調べるステップが、
前記データの新たなチェックサム値を計算するステップと、
前記新たなチェックサム値と前記チェックサム値とを比較するステップと備えた、請求項４３に記載のプログラム記憶装置。
【請求項４６】前記一貫性を保証するステップがさらに、
前記バッファに、前記第１のカプラーが書き込みアクセス可能であるとともに前記第２のカプラーが読み取りアクセス可能な第１の一連ファイルを準備するステップと、
前記バッファに、前記第１のカプラーが読み取りアクセス可能であるとともに前記第２のカプラーが書き込みアクセス可能な第２の一連ファイルを準備するステップと、
所定の順序で、前記第１のカプラーが前記第１の一連ファイルに書き込み、前記第２のカプラーが前記第１の一連ファイルから読み取るステップと、
前記所定の順序で、前記第２のカプラーが前記第２の一連ファイルに書き込み、前記第１のカプラーが前記第２の一連ファイルから読み取るステップとを備えた、請求

項４２に記載のプログラム記憶装置。

【請求項４７】前記インタフェースさせるステップがさらに、前記バッファ中のデータを監視して監視統計を供給するステップを備え、

前記監視統計が前記第１のサブモデルおよび前記第２のサブモデルを統計的に再分割して前記一括処理を最適化するのを容易にする、請求項３４に記載のプログラム記憶装置。

【請求項４８】前記モデルがＶＨＤＬで表されている、請求項３４に記載のプログラム記憶装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、一般にモデル・シミュレーションに関し、特に分割されたモデルのサブモデル群をコンカレント分散サブモデル・シミュレーション実行用に緊密かつ機能的に結合する手法に関する。

【０００２】

【従来の技術】モデル・シミュレーションは実システムを設計しその性能を予測する際における相変わらず重要な「ツール」である。モデル・シミュレーションは新たなシステムの適用による性能予測を実装前に生成するのにも役立つ。間違ったシステムを設計する危険性がきわめて大きい複雑なシステムでは、モデル・シミュレーションは必須である。

【０００３】分散シミュレーションとは、手元のシステムを様々なコンピュータ・ワークステーションに割り当てられ構成要素をなすサブモデルに分割して実行するシミュレーション環境のことである。

【０００４】一般に、分散シミュレーションでは、システム・サブモデル群を実行している協働ワークステーション群は最小限、各ワークステーションが実行すべき次のイベントに加え当該実行用に予定された時間を指示する、タイムスタンプを押されたイベント情報を交換する必要がある。従来技術では、分散シミュレーション用に２種類のモデルを用いている。すなわち、（ａ）「オプティミスティック（楽観的）」モデルと（ｂ）「コンサーバティブ（保守的）」モデルである。

【０００５】オプティミスティック・モデルでは、様々なプロセッサが処理するイベントの実行を順序付けない。その代わり、オプティミスティック・モデルでは、各ワークステーションは自分自身のイベント・シーケンスを別のワークステーションが処理するイベントとは無関係であるとみなして実行することができる。同時に、オプティミスティック・モデルには、因果関係エラーの検出とロールバックによって行なう引き続く復旧のための機構が実装されている。

【０００６】他方、コンサーバティブ・モデルは様々なワークステーションが実行するイベント間の相互依存関係を特定する先読みアルゴリズムを実装することにより、因果関係エラーを完全に排除するという観点に立っている。したがって、コンサーバティブ・モデルでは、あるワークステーションがイベントを処理できるのは、問題のイベントが、並行して処理されているイベントまたは他のワークステーションが次に処理するはずのイベントの結果に影響されないことが分かっている場合のみである。

【０００７】オプティミスティック手法の欠点のうちで最も顕著なものは、必要とされる検出・ロールバック機構がきわめて複雑であり実装するのが困難であるという事実である。

【０００８】コンサーバティブ・モデルは、上述したオーバーヘッドおよび不安定性とは無縁であるけれども、良好な性能を達成するためにはイベントの並列性を特定して利用する効率的で複雑な先読みアルゴリズムを必要とする。多くのアプリケーションがそのように効率的なアルゴリズムを開発できていない点を考えると、これは深刻な問題である。コンサーバティブ・モデルに付随する別の問題は、ユーザはシミュレートしている特定のアプリケーション／実験用のモデルを「微調整」できるように、使われているイベント同期方式を詳しく知っている必要があるという事実に関わっている。

【０００９】上述した２種類のモデルに共通する別の欠点は、（既製の市販ソフトウェアと異なり）特定の研究用または特定のアプリケーション・プログラム用に開発された特別のソフトウェアに依存している点である。さらに別の共通の欠点は、すべてのワークステーションが同じシミュレータのインスタンスを実行する必要があるという点である。

【００１０】上述した観点から、同じまたは異なるシミュレータのインスタンスを実行しているワークステーション群が通信するのを可能にするとともに、単純な検出・ロールバック機構を実現できる一般的な機構が当技術分野で求められている。

【００１１】

【課題を解決するための手段】本発明は一側面において、分割されたモデルの複数のサブモデルを処理する方法を提供することにより、従来技術の欠点を克服しさらなる利点を提供するものである。この方法はたとえば各々が対応するサブモデルにインタフェースした複数のカプラーを用いて上記複数のサブモデルを結合するステップと、上記複数のサブモデルを一括して処理するステップとを備え、上記複数のサブモデルが上記複数のカプラーおよび共通バッファを用いて通信するものである。

【００１２】本発明の別の側面では、上記複数のサブモデルを一括して処理するステップが、様々なシミュレータの複数のインスタンスを用いシミュレーション・プロセスの目標に従って各サブモデルを個別に処理しうるようにするステップを備えている。

【００１３】本発明のさらに別の側面では、上記バッファ中のデータが正確であるか否かを自動的に調べ、必要

な場合には再送信する。

【００１４】ここでは、上述した方法に対応するシステムとコンピュータ・プログラム製品についても述べられている。

【００１５】本発明の諸側面によれば、共有カプラー・ライブラリとライセンスを受けたハードウェア・シミュレータのＡＰＩ（アプリケーション・プログラミング・インタフェース）とを用いることにより、ライセンスを受けたハードウェア・シミュレータのソース・コードにわざわざアクセスしなくともモデルの分散処理を効果的に行なえる。本発明によれば、サブモデルが互いに直接に通信しうるようにすることにより、ハブレス（中心なし）分散処理、すなわち分散制御による処理が可能になる。さらに、ここに提示するモデル処理手法はコンピューティング・ユニット、そのアーキテクチャ、そのオペレーティング・システム、およびライセンスを受けたハードウェア・シミュレータで使われるプログラミング言語に関し特定の要件すなわち限定を必要としない。これは、共有カプラー・ライブラリの汎用機能を用いライセンスを受けたハードウェア・シミュレータのＡＰＩと直接に通信することにより、かつ、様々なアーキテクチャおよび／またはオペレーティング・システムの特定の機能に依存しないことにより実現されている。（「Ａおよび／またはＢ」は「ＡおよびＢ、Ａ、またはＢ」を表わす。）

【００１６】本発明の手法によれば、さらなる機能と利点を実現しうる。本発明の他の実例と側面をここで詳細に説明するが、それらは特許請求の範囲に記載した本発明の一部と考えられる。

【００１７】

【発明の実施の形態】本発明の少なくとも１つの側面により、複数のサブモデルに分割された１つのモデルを一括して処理する手法を提示する。この手法によれば、たとえば同じライセンスを受けたハードウェア・シミュレータまたは異なるライセンスを受けたハードウェア・シミュレータまたは複数のサブモデルを処理するライセンスを受けたプログラムのインスタンス群を接続する汎用通信機構を備えることによりモデルを処理するのが容易になる。

【００１８】以下、本発明の諸側面を取り込んで使用している、分散シミュレーション・システムの一実施形態を図１を参照して説明する。この分散コンピューティング環境はライセンスを受けたハードウェア・シミュレータのインスタンス１０４を実行するコンピューティング・ユニット１００と、ライセンスを受けたハードウェア・シミュレータのインスタンス１０６を実行するコンピューティング・ユニット１０２とを備えている。各コンピューティング・ユニットは、たとえばインターナショナル・ビジネス・マシーンズ・コーポレーション（ＩＢＭ）が提供するＵＮＩＸ(R)ベースのオペレーティング・システムであるＡＩＸを実行するＩＢＭ　ＲＩＳＣ／６０００コンピュータである。一実施形態では、インスタンス１０４、１０６は異なるライセンスを受けたハードウェア・シミュレータ、たとえばモデル・テクノロジー社が提供するＶＳＩＭやインターナショナル・ビジネス・マシーンズ・コーポレーションが提供するＰＳＩＭなどのインスタンスである。しかし、別の実施形態では、インスタンス１０４、１０６は同じライセンスを受けたハードウェア・シミュレータのインスタンスであってもよい。

【００１９】インスタンス１０４はカプラー１１０がインタフェースしているサブモデル１０８を処理する。（「インタフェースする」とはある構成要素が異質の他の構成要素に接続することである。）一方、インスタンス１０６はカプラー１１４がインタフェースしているサブモデル１１２を処理する。サブモデル１０８、１１２は処理すべきハードウェアのモデルを分割することにより生成する。様々な分割手法のさらなる詳細は文献や本願の同時米国出願に述べられている。ハードウェア・モデル分割の一例を下で図２（ａ）（ｂ）（ｃ）を参照して説明する。一方、本発明の一側面に係る汎用カプラーの一実施形態、および当該カプラーでサブモデルにインタフェースする方法を下で図３を参照して説明する。

【００２０】図１の参照を続ける。カプラー１１０、１１４にはバッファ１１６がインタフェースしている。バッファ１１６は一般に、カプラー１１０、１１４がアクセス可能でサブモデル１０８、１１２が通信するのを可能にする通信媒体を備えている。一例を挙げると、この通信媒体にはＩＢＭが提供しているＡＦＳ（Andrew File System）を用いた記憶装置を用いることができる。ＡＦＳとは、協働しているホスト群がＬＡＮ（local area network）とＷＡＮ（widw area network)の双方をまたいでファイルシステム資源を効率的に共有するのを可能にする分散ファイルシステムのことである。この特徴により、本発明でＡＦＳを使用することができる。さらに、上記記憶装置には共通通信ディレクトリ（ＣＣＤ）を備えることができる。このＣＣＤはカプラー１１０、１１４が読み書きアクセス可能な複数ファイル１１８を含んでいる。一実施形態では、この複数ファイルはモデルまたは分割されたサブモデルの処理が始まる前にＣＣＤ中に存在している。複数ファイル１１８はサブモデル１０８と１１２の間でカプラー１１０と１１４を介してデータを転送するのに使うことができる。一例では、カプラー１１０がサブモデル１０８からデータを取得し、それを複数ファイル１１８のうちの少なくとも１つのファイルに書き込む。次いで、カプラー１１４がこのファイルからデータを読み取り、当該データをサブモデル１１２に渡す。サブモデル１１２からサブモデル１０８へのデータの転送も同様の方法で行なう。一例として、ファイルにデータを書き込むのに用いる論理の一実施形態

を下で図７を参照して詳述し、ファイルからデータを読み取るのに用いる論理の一実施形態を下で図８を参照して詳述する。

【００２１】一実施形態では、複数ファイル１１８は２つのファイルから構成することができる。この実施形態では、一方のファイルはサブモデル１０８からサブモデル１１２にデータを転送するのに使用することができ、他方のファイルはサブモデル１１２からサブモデル１０８にデータを転送するのに使用することができる。

【００２２】別の実施形態では、入出力速度を速めるために、複数のサブモデル間でデータを転送するのに一連ファイルを用いることができる。一例として、一連ファイルに所定の順序でデータを書き込むことができる。次いで、このデータは当該一連ファイルから同じ所定の順序で読み取ることができる。したがって、書き込みの間の間隔が保証されているから、データをバッファに書き込もうとしているカプラーは当該バッファ中のファイルが上書きされないかどうかを調べる必要がない。一例では、一連ファイルは決定論的なラウンド・ロビン方式で読み書きされる。任意のカプラーが任意のファイルを再書き込みしうるようになるには、その前にサブモデルの一括処理全体がデータの交換を多数回結合することにより先行している必要がある。これにより、より速く処理されるサブモデルが待機することなく少なくともいくつかのデータを出力しうることが保証される。同様に入力側では、より遅く処理されるサブモデルが、読み取る入力データを取得することが保証される。

【００２３】上述したシステムとコンピューティング・ユニットは例として提示しただけである。本発明の通信インタフェースは、本発明の本旨のうちで、様々な種類のコンピューティング・ユニット、コンピュータ、プロセッサ、ノード、システム、ワークステーション、および／または環境に内蔵する、あるいはそれらとともに使用することができる。たとえば、コンピューティング・ユニットのうちの少なくとも１つはＵＮＩＸ(R)アーキテクチャに基づいていてもよいし、あるいはインテルＰＣアーキテクチャを内蔵していてもよい。さらに、ここで説明するいくつかの実施形態は２つのサブモデルを処理する２つのコンピューティング・ユニットしか備えていないが、任意個数のサブモデルを処理する任意個数のコンピューティング・ユニットを用いることができる。さらに、通信媒体としては、ここで説明する通信プロトコルを順守する任意の型の通信媒体、たとえばパケットまたはメッセージの送受信を介したネットワークなどを用いることができる。また、他の型のシステムも本発明の利益を享受できるから、本発明の一部をなすと考えられる。さらに、上述した方法には、オペレーティング・システム、あるいは、ライセンスを受けたハードウェア・シミュレータもしくはインスタンス中のライセンスを受けたプログラムで使用されている、またはそれに潜在するプログラミング言語に対する特定の要件は付随しない（すなわちオペレーティング・システムまたはプログラミング言語として任意のものを使用することができる）。

【００２４】以下、上述したように、本発明の機能を用いてシミュレーションを実行するあるモデルの一実施形態を図２（ａ）（ｂ）（ｃ）を参照して説明する。未分割のモデル２００は、たとえば動作コンポーネント２０２とクロック／サイクル・エンティティ２０４を備えている。クロック／サイクル・エンティティ２０４は動作コンポーネント２０２が変化しうる時を決める。

【００２５】ＡＳＩＣ（特定用途向け集積回路）のチップ設計概念では、モデルの動作コンポーネントはラッチ、ゲート、および配線を備えている。一方、変化画定エンティティはクロックが分布した配線上のクロック波形値である。動作コンポーネントは新たな値を取得し、クロック波形のサイクルに基づいて新たな値を出力する。イベント・ドリブン用途ではクロック・イベント、たとえば波形の立ち上がりを利用する。一方、サイクル・ドリブン用途ではクロック／サイクル遷移を利用する。上記２種類の用途はそれぞれのエンティティを用いてモデルの動作コンポーネントが変化しうる時を決める。

【００２６】一実施形態では、モデルは、たとえば図２（ｂ）に詳細に示すサブモデル２０６、２０８のように、内部で互いに通信している複数のサブモデルを備えているように概念的に説明することができる。サブモデル２０６、２０８の動作コンポーネントは入出力２１８を備えているように表わすことができる。入出力２１８はサブモデル間のデータ交換用に使われる。クロック／サイクル・エンティティ２１４は両サブモデルに共通であるが、サブモデル２０６と２０８の間に分布しており、チャネル２１６を介して結合されているように表わすことができる。モデルはサブモデルに分割することができる。それには、サブモデル２０６の入力をサブモデル２０８の出力から分離するとともにサブモデル２０８の入力をサブモデル２０６の出力から分離する。そして、サブモデル間にまたがるクロック／サイクル・エンティティ２１４をチャネル２１６のところで分割する。

【００２７】モデル２００のサブモデルへの概念的な分割を図２（ｃ）に示す。分割の目的は各々が入力２２２、２２４と出力２２０、２２６を備えたサブモデルを生成することであり、分割に由来するすべてのサブモデルに共通のクロック／サイクル・エンティティを実現することでもある。実現されたクロック／サイクル・エンティティは各サブモデルにおいてクロック／サイクル・ポート２２８、２３０として表わすことができる。

【００２８】実際の分割は良く定義されたインタフェースを目標とする人手によるプロセス、またはインタフェースを構築する際に依拠するアルゴリズム・プロセスに

なる可能性がある。上述した同時特許出願には、自動分割プロセスの例が記載されている。

【００２９】本発明の一側面として、ここでは、各サブモデルにインタフェースしてたとえば異なるコンピューティング・ユニットで実行されているサブモデル間の通信を可能にする汎用カプラーを提示する。そのようなカプラーの一例、およびインタフェース動作の一例を図３に示す。

【００３０】一実施形態として、サブモデル３００をＶＨＤＬ(VHSIC (Very High Speed Integrated Circuits) Hardware Description Language) を用いて表わす。ＶＨＤＬはＩＥＥＥ (Institute of Electrical and Electronics Engineers)が開発した標準 (VHDL-1076)である。ＶＨＤＬにおけるサブモデルは、ここで用いているように、少なくとも１つの出力３１０、少なくとも１つ入力３１２、およびクロック／サイクル・ポート３１４を備えたエンティティである。この実施形態では、カプラー３０２はトップ・レベル・カプラー・エンティティ３０４を備えている。そして、トップ・レベル・カプラー・エンティティ３０４自体がＶＨＤＬトップ・レベル・エンティティと汎用ＶＨＤＬカプラー・エンティティを備えている。一接続形式では、ＶＨＤＬトップ・レベル・エンティティを用いてサブモデルの入力、出力、およびクロック／サイクル・ポートを汎用ＶＨＤＬカプラー・エンティティの標準ポートに接続することができる。標準ポートには出力ポート３１６、入力ポート３１８、およびクロック／サイクル・ポート３２０がある。

【００３１】ＶＨＤＬトップ・レベル・エンティティ内では、サブモデルと汎用ＶＨＤＬカプラー・エンティティをインスタンス化したのち、ポート・マップを介してトップ・レベルの信号に接続する。サブモデル３００の一例を図４に示す。汎用ＶＨＤＬカプラー・エンティティの一例を図５に示す。そして、ＶＨＤＬトップ・レベル・エンティティ、およびトップ・レベル・カプラー・エンティティ３０４とサブモデル３００との間の接続形式を図６に示す。

【００３２】図３の参照を続ける。トップ・レベル・カプラー・エンティティ３０４の入力ポート３１８と出力ポート３１６はサブモデル３００の入力３１２と出力３１０にそれぞれマップされている。これにより、カプラー３０２とサブモデル３００との間の通信が可能になる。クロック／サイクル・ポート３２０はサブモデル３００のクロック／サイクル・ポート３１４にマップされている。これにより、カプラー３０２がサブモデル３００のクロック／サイクル・エンティティを検出するのが可能になる。カプラー３０２はクロック／サイクル・ポート３１４の信号を利用して、出力３１０から別のサブモデルへ送信するデータを集めたり、ＣＣＤから入力３１２へデータを渡したりする。さらに、カプラー３０２はカプラー共有ライブラリ３０８を備えている。カプラー共有ライブラリ３０８には、サブモデルが入出力するデータ・フローをカプラー３０２が制御するのを可能にする機能が含まれている。

【００３３】一実施形態では、クロック／サイクル・ポート３１４で変化が検出されると、カプラー共有ライブラリ３０８が呼び出される。カプラー３０２はカプラー共有ライブラリ３０８が備えている機能（図７～図１０参照）を使って、サブモデルから出力値を集めその情報を出力送信先たとえばＣＣＤに渡す、あるいは、入力源たとえばＣＤＤから入力情報を集めその情報をサブモデルに渡す。

【００３４】一例では、カプラー共有ライブラリはＣプログラミング言語を用いて実装することができる。この実施形態では、カプラー３０２はさらに、トップ・レベル・カプラー・エンティティとカプラー共有ライブラリをインタフェースする、ＶＨＤＬへのＡＰＩ (application programming interface) ３０６を備えている。

【００３５】以下、カプラー共有ライブラリが備え、カプラーがＣＣＤからデータを入力するとき、およびＣＣＤへデータを出力するときに使ういくつかの論理を図７～図１０を参照して詳細に説明する。

【００３６】あるサブモデルから別のサブモデルにデータを送信するには、送信すべきデータをまずＣＤＤに出力する。一実施形態では、このデータを図７に示す論理に従ってファイルに書き込む。

【００３７】まず、カプラーが書き込むべきデータを検索・取得、すなわちサブモデルからデータを取得する（５００）。一実施形態では、カプラーがこのデータのコピーをローカル記憶装置（図示せず）に格納する。この格納したデータは再送信機能を実行する次のサイクルで必要になる可能性がある。次いで、カプラーは書き込み用のあるファイルを読み取り用にオープンし（５０２）、当該ファイルを読み取って再送信要求がないかどうか各レコードを調べる（５０４）。再送信要求があるとしたら、それは上記ファイル中のデータが正しくないことを見いだした先行読み取り者が上記ファイルに書き込んだものである。ところで、高信頼性環境では、データを再送信する機能はシステム全体を高速化するために任意実行事項として無効にしうる。再送信要求をファイルに書き込むのに使う論理の例を下で図１０を参照して詳述する。

【００３８】図７の参照を続ける。再送信要求が存在する場合（５０６）、カプラーは再送信が有効であるか否かを調べる（５２０）。Ｙｅｓの場合、ファイルをクローズし、カプラーは再送信機能を呼び出す（５２６）。再送信機能の一実施形態は下で図８を参照して詳述する。再送信機能は先行サイクルでローカル記憶装置に格納したデータをカプラーに供給し、ファイルへのデータの書き込みを直ちに継続する（５１４）。データの再送信が有効でない場合、ファイルをクローズし、カプラー

は書き込み手順を終了する（５２４）。一実施形態では、発生した問題点を記載したログ・ファイルを生成してもよい。

【００３９】再送信要求が存在しない場合、カプラーはレコードにＩＮＶＡＬＩＤ（無効）キャラクタがないかどうか調べる（５０８）。すべてのレコードにこのキャラクタが存在しない場合、このファイルは最後まで読み取られていないから、上書きしてはいけない。この場合、カプラーはファイルをクローズし（５１０）、書き込み手順を初めから開始する（５０２）。すべてのレコードがＩＮＶＡＬＩＤとマークされている場合には、当該ファイルは上書きできる状態にある。カプラーは読み取り用の当該ファイルをクローズしたのち、それを書き込み用にオープンする（５１２）。

【００４０】次に、カプラーはデータを当該ファイルに書き込む（５１４）。一実施形態では、カプラーは当該ファイルの各レコードにＶＡＬＩＤ（有効）キャラクタ、順序レコード番号、正確かつ同一のサイクル番号、およびレコード固有データを書き込む。サイクル番号はデータに付随するクロック・サイクルを特定するものである。順序レコード番号は一実施形態では、当該ファイル中の各レコードの序数を表しており、エラー検出用に使うことができる。チェックサムを計算し、当該ファイルの最終レコードにレコード固有データとして書き込むのが望ましい（５１６）。一実施形態では、チェックサムの計算は当該ファイルのすべてのレコードに排他的ＯＲ操作を１６進形式で適用することにより行なっている。このあと、当該ファイルをクローズした（５１８）のち、書き込み手順は終了する（５２８）。

【００４１】以下、カプラーが再送信機能を実行するのに使用する論理の一実施形態を図８を参照して説明する。

【００４２】まず、カプラーは書き込み用にファイルをオープンする（６０２）。次いで、カプラーは先行サイクルで格納されたデータを取得し（６０４）、それを上記ファイルに書き込む準備をする（６０６）。このあと、カプラーは書き込み手順を開始する。

【００４３】データ再送信の第２の部分はＣＣＤからデータを検索・取得して、付随するカプラーがサブモデルに入力するのに備えることである。一実施形態では、このデータはたとえば図９に示す論理に従ってファイルから読み取る。

【００４４】まず、カプラーがクロック／サイクル・ポートを介してそのサブモデルがデータを必要としていることを検出したのち、読み取り手順を開始する（７００）。カプラーは上記サブモデルに入力すべきデータを格納しているＣＣＤ中のファイルを読み取り用としてオープンする（７０２）。次いで、カプラーが当該データを上記ファイルからロードする（７０４）。

【００４５】次に、ＶＡＬＩＤキャラクタ、正確な順序レコード番号、および正確かつ同一のサイクル番号をあるかどうか上記ファイルのレコード群を調べる（７０８）。あるサブモデルが他のサブモデルよりも速く処理される可能性があるから、上記ファイル中のデータが読み取りに適した状態にないことがありうる。もしも、あるレコード中に不正確なデータが存在する場合には、当該ファイルをクローズした（７１０）のち、読み取り手順を繰り返す（７０２）。

【００４６】すべてのレコードが読み取りに適した状態にあるとともに正確なデータを含んでいる場合、カプラーは当該すべてのレコードに対して１６進形式で累積的排他ＯＲを計算することによりチェックサム値を算出する（７１２）。ただし、最終レコードはチェックサム値を格納しているから、この計算から除外する。次いで、算出したチェックサム値と最終レコードに格納されているチェックサム値とを比較する（７１４）。

【００４７】２つのチェックサム値が一致しない場合、カプラーはエラーを報告した（７１６）のち、再送信が有効であるか否かを調べる（７１８）。Ｙｅｓの場合、当該読み取り用ファイルをクローズした（７２０）のち、カプラーは再送信要求手順を開始する（７２２）。再送信要求機能で使われる論理の一例を下で図１０を参照して説明する。再送信を要求したのち、読み取り手順を繰り返す（７０２）。読み取り手順は次の書き込み者が再送信を実行するまで繰り返すことになる。再送信が有効でない場合には、カプラーは当該ファイルをクローズしたのち、デバッグ・ログ機能を実行する（７２４）。

【００４８】２つのチェックサム値が一致する場合、カプラーは次のサイクルのために最終チェックサム値の格納を開始する。次いで、当該ファイルを読み取り用にはクローズしたのち、書き込み用にオープンして（７３０）、当該ファイルの将来の読み取り者に当該ファイルがすでに読み取られていることを知らせることができるようにする。そうするために、カプラーは各レコードにＩＮＶＡＬＩＤ（無効）とマーク付ける。次いで、当該ファイルをクローズした（７３４）のち、カプラーは読み取り手順を終了する（７３６）。この時点で、カプラーは当該ファイルから読み取ったデータを保持しているから、それをサブモデルに当該サブモデルの入力を介して渡すことができる。

【００４９】以下、カプラーが再送信を要求するのに使用しうる論理の一例を図１０を参照して説明する。

【００５０】カプラーが読み取りプロセスの間にチェックサム・エラーに遭遇すると、図１０の手順が開始する（８００）。まず、カプラーはファイルを書き込み用にオープンし（８０２）、当該ファイルの各レコードに再送信要求を書き込んだ（８０４）のち、当該ファイルをクローズする（８０６）。これにより、次の書き込み者は再送信機能を（図７、図８に示したように）実行すべ

きことを知らされることになる。以後、カプラーは読み取り手順を開始する。

【００５１】一実施形態では、処理システムの信頼性が十分に高い場合には、当業者にとって明らかなように、ファイルの一貫性を保証する論理を上述したプロセス・フローから取り除いてシステム全体の速度を高めることができる。そのような実施形態では、書き込み手順は検査プロセスを含んでおらず、読み取り手順はサイクル番号しか検査しない。速く処理されるサブモデルは遅く処理されるサブモデルを待つとともに、集合体の同期性を維持するためにファイルを再読み取りすることを強制される。これに対して、最も遅いサブモデルはつねに入力に適した状態の入力データを取得するとともに、再読み取りは全く行なわない。したがって、再読み取り回数は処理コンピューティング・ユニットに課される負荷の尺度である。たとえば、再読み取り回数が少ないということは負荷が大きい、すなわちコンピューティング・ユニットが遅いということを表わしている。

【００５２】一実施形態では、シミュレーション実行間でサブモデルを静的に再分割するのに再読み取り回数の統計を利用することができる。別の実施形態では、たとえば遅いアプリケーション・インスタンスの優先順位を引き上げることにより、コンピューティング資源をこの遅いアプリケーション・インスタンスに動的に割り当てるのに再読み取り回数の統計を利用することができる。この実施形態では、１つの遅いポイント（遅いインスタンス）をインスタンスの集合と取り替えても、モデル集合体としては計算が進行する速度が低下することはない。さらに別の実施形態では、ある集合体中のすべてのコンピューティング・ユニットにわたる再読み取り回数の統計は、当該集合体全体の性能の尺度として利用することができる。

【００５３】さらに別の実施形態では、ファイルに書き込むのに適した状態にあるデータを連結して１つのレコードにすることができるから、当該データを出力するのにたった１回の書き込みしか必要としない。同様に、ファイルから読み取るのに適した状態にあるデータを当該ファイルから入力するには、たった１回の読み取りしか必要としない。たった１回の読み取りでファイルから取得したデータは、入力側で分解することができる。

【００５４】さらに別の実施形態では、サブモデルを処理するハードウェア・シミュレータまたはアプリケーション・プログラムの各インスタンスを一連のコンピュータにわたり非同期に開始することができる。各インスタンスが先に進むのを止められるのは、入力資源または出力資源がそれを強制的に待たせる場合のみである。このように、インスタンス群は全体として非接続で（すなわち互いに無関係に）開始することができ、集合体中の他の構成要素がＣＣＤ中のファイルを更新するまでずっと先に進むことができる。同様に、集合体全体を所定の時点まで進ませる、あるいは所定の時点で停止させるのに、キーボード入力を用いたフォアグラウンド（前景）セッションを利用することができる。

【００５５】ここで開示した結合機構によって、最も遅いサブモデルのインスタンスは、より速いインスタンスから必要な情報を取得するのに自身が費やす時間を最小限にすることにより、可能な限り最大の速度で実行することが可能になる。より速いインスタンスは情報を待つように強制されるとともに、自身が継続しうるように次に必要な情報が取得可能であるか否かを判断するのに自身のコンピューティング資源を費やすように強制される。これにより、再計算を伴う予測と引き続くバックアップに資源を費やすのを避けることが可能になる。

【００５６】以上、本発明に従いモデル処理の様々な側面を説明した。分割されたモデルを分散処理する方法、および異なるコンピューティング・ユニットで処理されるサブモデルを結合かつ制御する手法を提示した。ここで説明した様々な手法は単一システム、同種システム、および異機種システムに適用することができる。一例として、同じ物理マシン上の異なるハードウェア・シミュレータ群によって複数のサブモデルを処理することができる。

【００５７】本発明はたとえばコンピュータ利用可能な媒体を備えた製品（たとえば少なくとも１つのコンピュータ・プログラム製品）中に含めることができる。上記媒体はその中にたとえば本発明の機能を提供するとともに容易にするコンピュータ読み取り可能なプログラム・コード手段を組み込んでいる。上記製品はコンピュータ・システムの一部に含めることもできるし、別売することもできる。

【００５８】さらに、本発明によれば、マシンによって読み取り可能であるとともに、マシンによって実行可能であり本発明の機能を実行する命令群から成る少なくとも１つのプログラムを有形的に記録した少なくとも１つのプログラム記憶装置を提供することができる。

【００５９】ここに示したフローチャートは単なる例にすぎない。本発明の本旨のうちで、これらのフローチャートまたはここで説明したステップ（すなわち操作）群には多くの変形例がありうる。たとえば、ステップ群は異なる順序で実行しうるし、あるいはステップを付加、削除、または変形することができる。これらの変形例はすべて特許請求の範囲の一部をなすものと考えられる。

【００６０】以上、ここでは好適な実施形態を示すとともに説明したけれども、当業者にとって明らかなように、本発明の本旨のうちで様々な変形、付加、置換などをなすことができる。したがって、これらのものは特許請求の範囲で定義された本発明の範囲の内にあると考えられる。

【００６１】まとめとして以下の事項を開示する。

（１）分割されたモデルの、第１のサブモデルおよび第

２のサブモデルを含む複数のサブモデルを処理する方法であって、分割されたモデルの第１のサブモデルと第２のサブモデルとを結合するステップであって、前記第１のサブモデルがインタフェースする第１のカプラーを準備するステップと、前記第２のサブモデルがインタフェースする第２のカプラーを準備するステップと、前記第１のカプラーと前記第２のカプラーとをインタフェースさせるステップであって、前記第１のカプラーと前記第２のカプラーとの間にバッファを準備するステップとを備えたステップと、前記第１のサブモデルおよび前記第２のサブモデルを一括して処理するステップであって、前記第１のサブモデルおよび前記第２のサブモデルが前記第１のカプラー、前記バッファ、および前記第２のカプラーを用いて通信する、ステップとを備えたステップを備えた方法。

（２）前記モデルが論理設計である、上記（１）に記載の方法。

（３）前記一括して処理するステップが前記第１のサブモデルおよび前記第２のサブモデルを分散処理するステップを備え、前記分散処理するステップが分散プロセス制御を備えている、上記（１）に記載の方法。

（４）前記一括して処理するステップがさらに前記第１のサブモデルを第１のシミュレータ・インスタンス上で処理するステップと前記第２のサブモデルを第２のシミュレータ・インスタンス上で処理するステップとを備え、前記第１のシミュレータ・インスタンスは前記第２のシミュレータ・インスタンスと異なり、前記第１のシミュレータ・インスタンスは前記バッファを用いて前記第２のシミュレータ・インスタンスと通信する、上記（１）に記載の方法。

（５）前記第１のシミュレータ・インスタンスは第１のコンピューティング・ユニットに常駐し、前記２のシミュレータ・インスタンスは第２のコンピューティング・ユニットに常駐する、上記（４）に記載の方法。

（６）少なくとも１つのサブモデルおよび付随するカプラーが少なくとも１つのマップされたデータ・ポートおよびクロック／サイクル・ポートを備え、前記クロック／サイクル・ポートは少なくとも１つのクロック／サイクル信号を供給し、前記少なくとも１つのクロック／サイクル信号は前記少なくとも１つのサブモデルと付随するカプラーとの間で前記少なくとも１つのマップされたデータ・ポートを通じてデータを転送するのに使用される、上記（１）に記載の方法。

（７）前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方がトップ・レベル・カプラー・エンティティとカプラー共有ライブラリとを備え、前記第１のカプラーおよび前記第２のカプラーのうちの前記少なくとも一方が前記トップ・レベル・カプラー・エンティティを用いて前記第１のサブモデルおよび前記第２のサブモデルと個別に接続し、前記トップ・レベル・カプラー・エンティティが少なくとも１つの入力と、少なくとも１つの出力と、前記第１のサブモデルおよび前記第２のサブモデルのうちの少なくとも一方の前記クロック／サイクル・ポートにマップされた複数のポートとを備え、前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方が前記カプラー共有ライブラリを用いて前記カプラーと前記バッファとの間でデータを転送する、上記（６）に記載の方法。

（８）前記バッファが通信媒体を備え、前記通信媒体は前記第１のカプラーと前記第２のカプラーとの間でデータを渡すとともに監視する少なくとも１つのプロトコルを備えている、上記（１）に記載の方法。

（９）前記一括して処理するステップがさらに、前記第１のカプラーが前記第１のサブモデルから取得したデータを前記バッファに送信し、前記第２のカプラーが前記データを前記バッファから取得し、前記データを前記第２のサブモデルに供給するステップと、前記送信と前記取得とを対応させることにより、前記データの一貫性を保証するステップとを備えた、上記（１）に記載の方法。

（１０）前記一貫性を保証するステップが、前記データにチェックサム値を挿入するステップと、前記第１のカプラーおよび前記第２のカプラーのうちの一方が前記バッファ中の前記データを調べて前記データの正確性を判断するステップと、前記データが不正確な場合、前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方が、前記データの前記バッファへの再送信および前記データの前記バッファからの再読み取りのうちの少なくとも一方を行なうステップとを備えた、上記（９）に記載の方法。

（１１）前記挿入するステップが、前記チェックサム値を計算するステップを備え、前記計算するステップが、少なくとも１つの論理操作を前記データに適用するステップを備えている、上記（１０）に記載の方法。

（１２）前記調べるステップが、前記データの新たなチェックサム値を計算するステップと、前記新たなチェックサム値と前記チェックサム値とを比較するステップと備えた、上記（１０）に記載の方法。

（１３）前記一貫性を保証するステップがさらに、前記バッファに、前記第１のカプラーが書き込みアクセス可能であるとともに前記第２のカプラーが読み取りアクセス可能な第１の一連ファイルを準備するステップと、前記バッファに、前記第１のカプラーが読み取りアクセス可能であるとともに前記第２のカプラーが書き込みアクセス可能な第２の一連ファイルを準備するステップと、所定の順序で、前記第１のカプラーが前記第１の一連ファイルに書き込み、前記第２のカプラーが前記第１の一連ファイルから読み取るステップと、前記所定の順序で、前記第２のカプラーが前記第２の一連ファイルに書き込み、前記第１のカプラーが前記第２の一連ファイル

から読み取るステップとを備えた、上記（９）に記載の方法。

（１４）前記インタフェースさせるステップがさらに、前記バッファ中のデータを監視して監視統計を供給するステップを備え、前記監視統計が前記第１のサブモデルおよび前記第２のサブモデルを統計的に再分割して前記一括処理を最適化するのを容易にする、上記（１）に記載の方法。

（１５）前記モデルがＶＨＤＬで表されている、上記（１）に記載の方法。

（１６）分割されたモデルの、第１のサブモデルおよび第２のサブモデルを含む複数のサブモデルを処理するシステムであって、分割されたモデルの第１のサブモデルと第２のサブモデルとを結合する手段であって、前記第１のサブモデルがインタフェースする第１のカプラーを準備する手段と、前記第２のサブモデルがインタフェースする第２のカプラーを準備する手段と、前記第１のカプラーと前記第２のカプラーとをインタフェースさせる手段であって、前記第１のカプラーと前記第２のカプラーとの間にバッファを準備する手段とを備えた手段と、前記第１のサブモデルおよび前記第２のサブモデルを一括して処理する手段であって、前記第１のサブモデルおよび前記第２のサブモデルが前記第１のカプラー、前記バッファ、および前記第２のカプラーを用いて通信する、手段とを備えた手段を備えた方法。

（１７）前記モデルが論理設計である、上記（１６）に記載のシステム。

（１８）前記一括して処理する手段が前記第１のサブモデルおよび前記第２のサブモデルを分散処理する手段を備え、前記分散処理する手段が分散プロセス制御を備えている、上記（１６）に記載のシステム。

（１９）前記一括して処理する手段がさらに前記第１のサブモデルを第１のシミュレータ・インスタンス上で処理する手段と前記第２のサブモデルを第２のシミュレータ・インスタンス上で処理する手段とを備え、前記第１のシミュレータ・インスタンスは前記第２のシミュレータ・インスタンスと異なり、前記第１のシミュレータ・インスタンスは前記バッファを用いて前記第２のシミュレータ・インスタンスと通信する、上記（１６）に記載のシステム。

（２０）前記第１のシミュレータ・インスタンスは第１のコンピューティング・ユニットに常駐し、前記２のシミュレータ・インスタンスは第２のコンピューティング・ユニットに常駐する、上記（１９）に記載のシステム。

（２１）少なくとも１つのサブモデルおよび付随するカプラーが少なくとも１つのマップされたデータ・ポートおよびクロック／サイクル・ポートを備え、前記クロック／サイクル・ポートは少なくとも１つのクロック／サイクル信号を供給し、前記少なくとも１つのクロック／サイクル信号は前記少なくとも１つのサブモデルと付随するカプラーとの間で前記少なくとも１つのマップされたデータ・ポートを通じてデータを転送するのに使用される、上記（１６）に記載のシステム。

（２２）前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方がトップ・レベル・カプラー・エンティティとカプラー共有ライブラリとを備え、前記第１のカプラーおよび前記第２のカプラーのうちの前記少なくとも一方が前記トップ・レベル・カプラー・エンティティを用いて前記第１のサブモデルおよび前記第２のサブモデルと個別に接続し、前記トップ・レベル・カプラー・エンティティが少なくとも１つの入力と、少なくとも１つの出力と、前記第１のサブモデルおよび前記第２のサブモデルのうちの少なくとも一方の前記クロック／サイクル・ポートにマップされた複数のポートとを備え、前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方が前記カプラー共有ライブラリを用いて前記カプラーと前記バッファとの間でデータを転送する、上記（２１）に記載のシステム。

（２３）前記バッファが通信媒体を備え、前記通信媒体は前記第１のカプラーと前記第２のカプラーとの間でデータを渡すとともに監視する少なくとも１つのプロトコルを備えている、上記（１６）に記載のシステム。

（２４）前記一括して処理する手段がさらに、前記第１のカプラーが前記第１のサブモデルから取得したデータを前記バッファに送信し、前記第２のカプラーが前記データを前記バッファから取得し、前記データを前記第２のサブモデルに供給する手段と、前記送信と前記取得とを対応させることにより、前記データの一貫性を保証する手段とを備えた、上記（１６）に記載のシステム。

（２５）前記一貫性を保証する手段が、前記データにチェックサム値を挿入する手段と、前記第１のカプラーおよび前記第２のカプラーのうちの一方が前記バッファ中の前記データを調べて前記データの正確性を判断する手段と、前記データが不正確な場合、前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方が、前記データの前記バッファへの再送信および前記データの前記バッファからの再読み取りのうちの少なくとも一方を行なう手段とを備えた、上記（２４）に記載の方法。

（２６）前記挿入する手段が、前記チェックサム値を計算する手段を備え、前記計算する手段が、少なくとも１つの論理操作を前記データに適用する手段を備えている、上記（２５）に記載のシステム。

（２７）前記調べる手段が、前記データの新たなチェックサム値を計算する手段と、前記新たなチェックサム値と前記チェックサム値とを比較する手段と備えた、上記（２５）に記載のシステム。

（２８）前記一貫性を保証する手段がさらに、前記バッファに、前記第１のカプラーが書き込みアクセス可能で

あるとともに前記第２のカプラーが読み取りアクセス可能な第１の一連ファイルを準備する手段と、前記バッファに、前記第１のカプラーが読み取りアクセス可能であるとともに前記第２のカプラーが書き込みアクセス可能な第２の一連ファイルを準備する手段と、所定の順序で、前記第１のカプラーが前記第１の一連ファイルに書き込み、前記第２のカプラーが前記第１の一連ファイルから読み取る手段と、前記所定の順序で、前記第２のカプラーが前記第２の一連ファイルに書き込み、前記第１のカプラーが前記第２の一連ファイルから読み取る手段とを備えた、上記（２４）に記載のシステム。

（２９）前記インタフェースさせる手段がさらに、前記バッファ中のデータを監視して監視統計を供給する手段を備え、前記監視統計が前記第１のサブモデルおよび前記第２のサブモデルを統計的に再分割して前記一括処理を最適化するのを容易にする、上記（１６）に記載のシステム。

（３０）前記モデルがＶＨＤＬで表されている、上記（１６）に記載のシステム。

（３１）分割されたモデルの、第１のサブモデルおよび第２のサブモデルを含む複数のサブモデルを処理するシミュレーション・システムであって、分割されたモデルの第１のサブモデルと第２のサブモデルとを、前記第１のサブモデルにインタフェースした第１のカプラーおよび前記第２のサブモデルにインタフェースした第２のカプラーを準備し、前記第１のカプラーと前記第２のカプラーとの間に設けたバッファによって前記第１のカプラーと前記第２のカプラーとをインタフェースさせることにより、結合するように適合した少なくとも１つのコンピューティング・ユニットを備え、前記少なくとも１つのコンピューティング・ユニットがさらに前記第１のサブモデルおよび前記第２のサブモデルを一括して処理するように適合しており、前記第１のサブモデルおよび前記第２のサブモデルが前記第１のカプラー、前前記バッファ、および記第２のカプラーを用いて通信するシミュレーション・システム。

（３２）前記少なくとも１つのコンピューティング・ユニットが複数のシミュレータ・インスタンスを備え、前記複数のシミュレータ・インスタンスが第１のシミュレータ・インスタンスおよび第２のシミュレータ・インスタンスを含み、前記第１のシミュレータ・インスタンスが前記第１のサブモデルを処理し、前記第２のシミュレータ・インスタンスが前記第２のサブモデルを処理し、前記第１のシミュレータ・インスタンスが前記第２のシミュレータ・インスタンスとは異なる、上記（３１）に記載のシミュレーション・システム。

（３３）前記第１のシミュレータ・インスタンスが第１のコンピューティング・ユニットに常駐し、前記第２のシミュレータ・インスタンスが第２のコンピューティング・ユニットに常駐する、上記（３２）に記載のシミュレーション・システム。

（３４）分割されたモデルの、第１のサブモデルおよび第２のサブモデルを含む複数のサブモデルを処理する方法であって、分割されたモデルの第１のサブモデルと第２のサブモデルとを結合するステップであって、前記第１のサブモデルがインタフェースする第１のカプラーを準備するステップと、前記第２のサブモデルがインタフェースする第２のカプラーを準備するステップと、前記第１のカプラーと前記第２のカプラーとをインタフェースさせるステップであって、前記第１のカプラーと前記第２のカプラーとの間にバッファを準備するステップとを備えたステップと、前記第１のサブモデルおよび前記第２のサブモデルを一括して処理するステップであって、前記第１のサブモデルおよび前記第２のサブモデルが前記第１のカプラー、前記バッファ、および前記第２のカプラーを用いて通信する、ステップとを備えたステップを備えた方法を実行する、機械によって実行可能な命令から成る少なくとも１つのプログラムを有形的に記録した、機械読み取り可能なプログラム記憶装置。

（３５）前記モデルが論理設計である、上記（３４）に記載のプログラム記憶装置。

（３６）前記一括して処理するステップが前記第１のサブモデルおよび前記第２のサブモデルを分散処理するステップを備え、前記分散処理するステップが分散プロセス制御を備えている、上記（３４）に記載のプログラム記憶装置。

（３７）前記一括して処理するステップがさらに前記第１のサブモデルを第１のシミュレータ・インスタンス上で処理するステップと前記第２のサブモデルを第２のシミュレータ・インスタンス上で処理するステップとを備え、前記第１のシミュレータ・インスタンスは前記第２のシミュレータ・インスタンスと異なり、前記第１のシミュレータ・インスタンスは前記バッファを用いて前記第２のシミュレータ・インスタンスと通信する、上記（３４）に記載のプログラム記憶装置。

（３８）前記第１のシミュレータ・インスタンスは第１のコンピューティング・ユニットに常駐し、前記２のシミュレータ・インスタンスは第２のコンピューティング・ユニットに常駐する、上記（３７）に記載のプログラム記憶装置。

（３９）少なくとも１つのサブモデルおよび付随するカプラーが少なくとも１つのマップされたデータ・ポートおよびクロック／サイクル・ポートを備え、前記クロック／サイクル・ポートは少なくとも１つのクロック／サイクル信号を供給し、前記少なくとも１つのクロック／サイクル信号は前記少なくとも１つのサブモデルと付随するカプラーとの間で前記少なくとも１つのマップされたデータ・ポートを通じてデータを転送するのに使用される、上記（３４）に記載のプログラム記憶装置。

（４０）前記第１のカプラーおよび前記第２のカプラー

のうちの少なくとも一方がトップ・レベル・カプラー・エンティティとカプラー共有ライブラリとを備え、前記第１のカプラーおよび前記第２のカプラーのうちの前記少なくとも一方が前記トップ・レベル・カプラー・エンティティを用いて前記第１のサブモデルおよび前記第２のサブモデルと個別に接続し、前記トップ・レベル・カプラー・エンティティが少なくとも１つの入力と、少なくとも１つの出力と、前記第１のサブモデルおよび前記第２のサブモデルのうちの少なくとも一方の前記クロック／サイクル・ポートにマップされた複数のポートとを備え、前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方が前記カプラー共有ライブラリを用いて前記カプラーと前記バッファとの間でデータを転送する、上記（３９）に記載のプログラム記憶装置。

（４１）前記バッファが通信媒体を備え、前記通信媒体は前記第１のカプラーと前記第２のカプラーとの間でデータを渡すとともに監視する少なくとも１つのプロトコルを備えている、上記（３４）に記載のプログラム記憶装置。

（４２）前記一括して処理するステップがさらに、前記第１のカプラーが前記第１のサブモデルから取得したデータを前記バッファに送信し、前記第２のカプラーが前記データを前記バッファから取得し、前記データを前記第２のサブモデルに供給するステップと、前記送信と前記取得とを対応させることにより、前記データの一貫性を保証するステップとを備えた、上記（３４）に記載のプログラム記憶装置。

（４３）前記一貫性を保証するステップが、前記データにチェックサム値を挿入するステップと、前記第１のカプラーおよび前記第２のカプラーのうちの一方が前記バッファ中の前記データを調べて前記データの正確性を判断するステップと、前記データが不正確な場合、前記第１のカプラーおよび前記第２のカプラーのうちの少なくとも一方が、前記データの前記バッファへの再送信および前記データの前記バッファからの再読み取りのうちの少なくとも一方を行なうステップとを備えた、上記（４２）に記載のプログラム記憶装置。

（４４）前記挿入するステップが、前記チェックサム値を計算するステップを備え、前記計算するステップが、少なくとも１つの論理操作を前記データに適用するステップを備えている、上記（４３）に記載のプログラム記憶装置。

（４５）前記調べるステップが、前記データの新たなチェックサム値を計算するステップと、前記新たなチェックサム値と前記チェックサム値とを比較するステップと備えた、上記（４３）に記載のプログラム記憶装置。

（４６）前記一貫性を保証するステップがさらに、前記バッファに、前記第１のカプラーが書き込みアクセス可能であるとともに前記第２のカプラーが読み取りアクセス可能な第１の一連ファイルを準備するステップと、前記バッファに、前記第１のカプラーが読み取りアクセス可能であるとともに前記第２のカプラーが書き込みアクセス可能な第２の一連ファイルを準備するステップと、所定の順序で、前記第１のカプラーが前記第１の一連ファイルに書き込み、前記第２のカプラーが前記第１の一連ファイルから読み取るステップと、前記所定の順序で、前記第２のカプラーが前記第２の一連ファイルに書き込み、前記第１のカプラーが前記第２の一連ファイルから読み取るステップとを備えた、上記（４２）に記載のプログラム記憶装置。

（４７）前記インタフェースさせるステップがさらに、前記バッファ中のデータを監視して監視統計を供給するステップを備え、前記監視統計が前記第１のサブモデルおよび前記第２のサブモデルを統計的に再分割して前記一括処理を最適化するのを容易にする、上記（３４）に記載のプログラム記憶装置。

（４８）前記モデルがＶＨＤＬで表されている、上記（３４）に記載のプログラム記憶装置。

【図面の簡単な説明】

【図１】　本発明の諸側面に従いサブモデル群を一括して処理する通信環境の一実施形態を示す図である。

【図２】　（ａ）本発明の一側面に従ってシミュレートする１つのモデルの一実施形態を示す図である。（ｂ）図２（ａ）のモデルの分割を示す図である。（ｃ）本発明の一側面に従い結果のサブモデルの一例を示す図である。

【図３】　本発明の一側面に従い汎用カプラーおよびそのサブモデルへのインタフェースの一例を示す図である。

【図４】　本発明の一側面に従いＶＨＤＬを用いて表したサブモデルの一例を示す図である。

【図５】　本発明の一側面に従いＶＨＤＬを用いて表した汎用カプラー・エンティティの一例を示す図である。

【図６】　本発明の一側面に従いＶＨＤＬトップ・レベル・エンティティの一例を示す図である。

【図７】　本発明の一側面に従いカプラー共有ライブラリが出力をファイルに書き込むのに使用する論理の一例を示す図である。

【図８】　本発明の一側面に従いカプラー共有ライブラリが再送信機能を実行するのに使用する論理の一例を示す図である。

【図９】　本発明の一側面に従いカプラー共有ライブラリがファイルから入力を読み取るのに使用する論理の一例を示す図である。

【図１０】　本発明の一側面に従いカプラー共有ライブラリがデータの再送信を要求するのに使用する論理の一例を示す図である。

【符号の説明】

１００　コンピューティング・ユニット

１０２　コンピューティング・ユニット

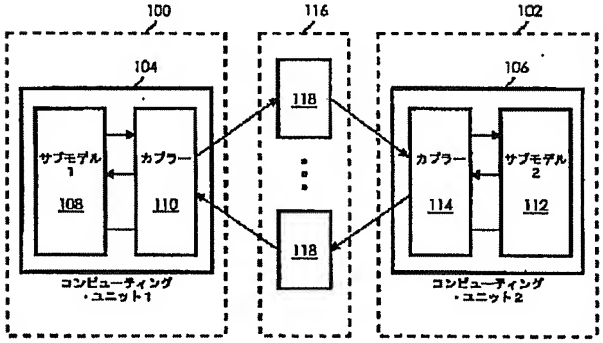
29

１０４　ライセンスを受けたハードウェア・シミュレータのインスタンス
１０６　ライセンスを受けたハードウェア・シミュレータのインスタンス
１０８　サブモデル
１１０　カプラー
１１２　サブモデル
１１４　カプラー
１１６　バッファ
１１８　複数ファイル
２００　未分割のモデル
２０２　動作コンポーネント
２０４　クロック／サイクル・エンティティ
２０６　サブモデル
２０８　サブモデル
２１０　サブモデル１の動作コンポーネント
２１２　サブモデル２の動作コンポーネント
２１４　クロック／サイクル・エンティティ
２１６　チャネル

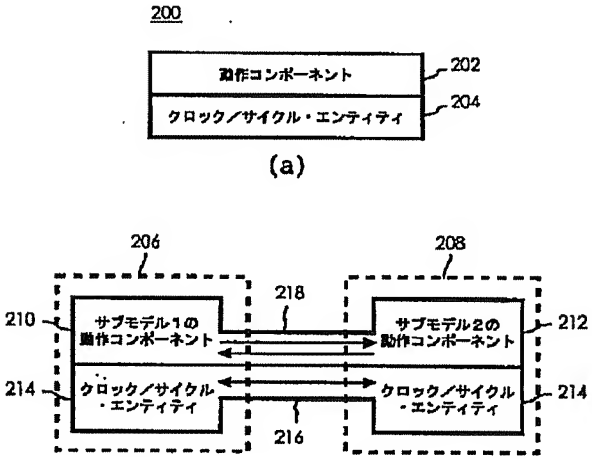
30

＊２１８　入出力
２２２　入力
２２４　入力
２２０　出力
２２６　出力
２２８　クロック／サイクル・ポート
２３０　クロック／サイクル・ポート
３００　サブモデル
３０２　カプラー
３０４　トップ・レベル・カプラー・エンティティ
３０６　ＶＨＤＬへのＡＰＩ
３０８　カプラー共有ライブラリ
３１０　出力
３１２　入力
３１４　クロック／サイクル・ポート
３１６　出力ポート
３１８　入力ポート
３２０　クロック／サイクル・ポート

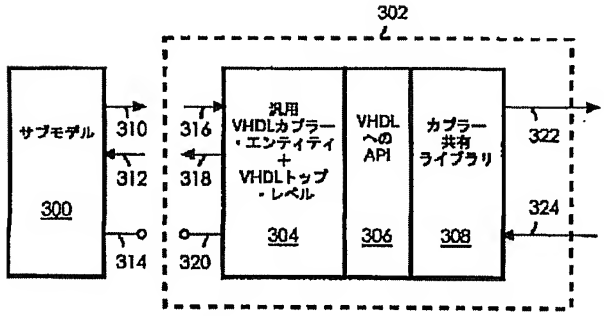
【図１】



【図２】



【図３】



【図４】

```
EXAMPLE OF VHDL SUBMODEL ENTITY:

    entity submodel is
        port (
            clock       : in std_ulogic;
            a_input     : in std_ulogic;
            b_output    : out std_ulogic;
            );
    end;
```

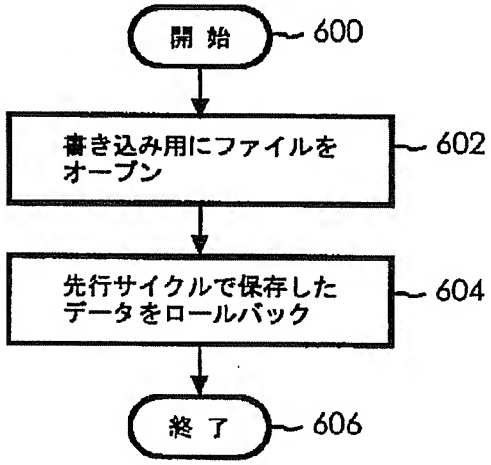
【図５】

```
EXAMPLE OF GENERIC COUPLER ENTITY:

    entity VHDL_Coupler_Entity is
        port (
            clock       : in std_ulogic;
            input_1     : out std_ulogic;
            output_1    : in std_ulogic;
            );
    end;
```

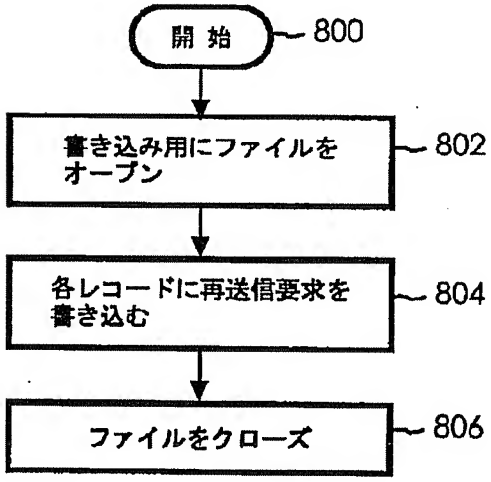
【図６】

```
EXAMPLE OF VHDL TOP LEVEL:

    entity VHDL_Top_Level to VHDL_Top_level is
    end;
    architecture VHDL_Top_Level of VHDL_Top_level is
        component submodel
            port (
                clock       : in std_ulogic;
                a_input     : in std_ulogic;
                b_output    : out std_ulogic;
                );
        end component;

        component VHDL_Coupler_Entity
            port (
                clock       : in std_ulogic;
                input_1     : out std_ulogic;
                output_1    : in std_ulogic;
                );
        end component;

        signal clock        : in std_ulogic;
        signal input_1      : out std_ulogic;
        signal output_1     : in std_ulogic;

        begin

        SUBMODEL : submodel
                port map (
                        clock       =>  clock,
                        a_input     =>  input_1,
                        b_output    =>  output_1
                        );

        VHDL_COUPLER_ENTITY  :  VHDL_Coupler_Entity
                port map (
                        clock,
                        input_1,
                        output_1
                        );

        end VHDL_Top_level;
```

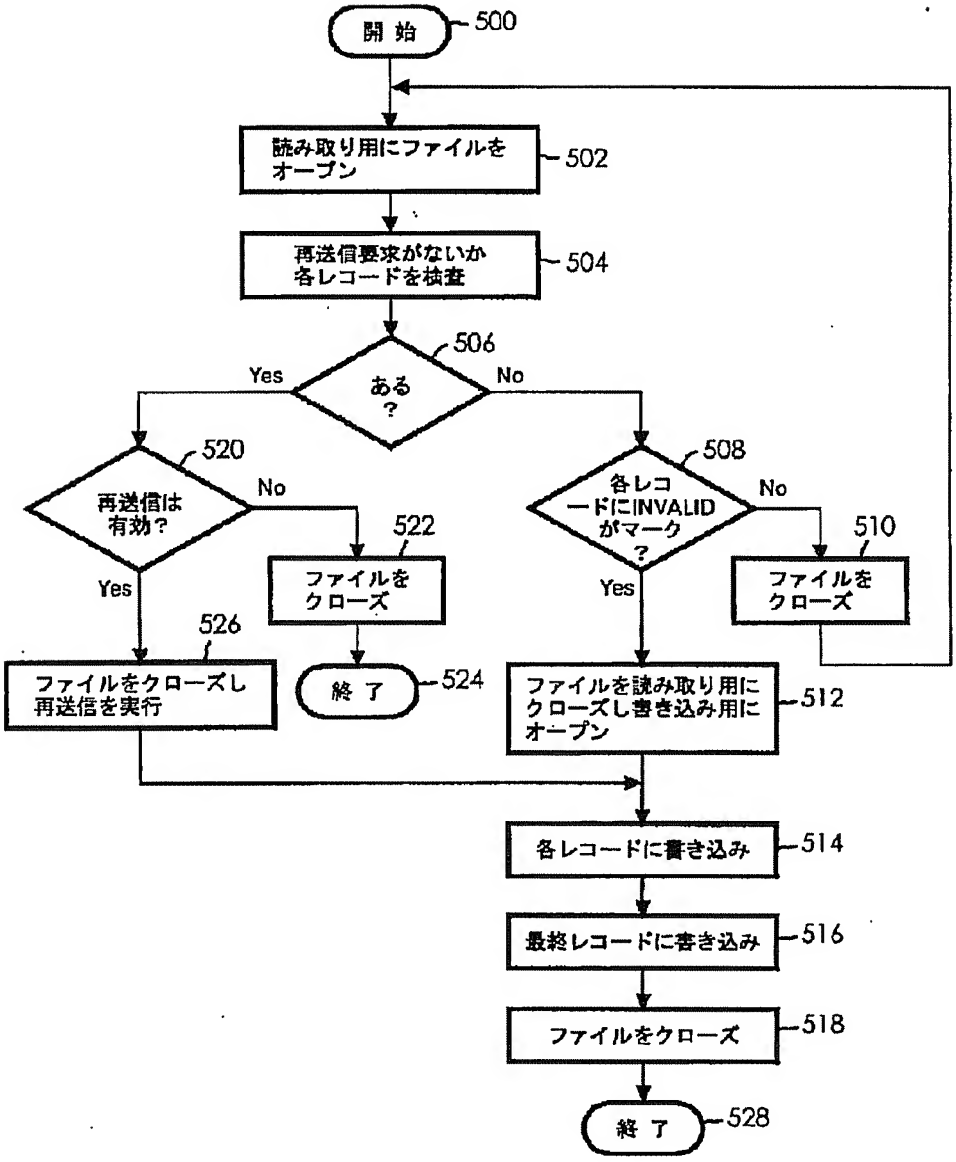
【図８】



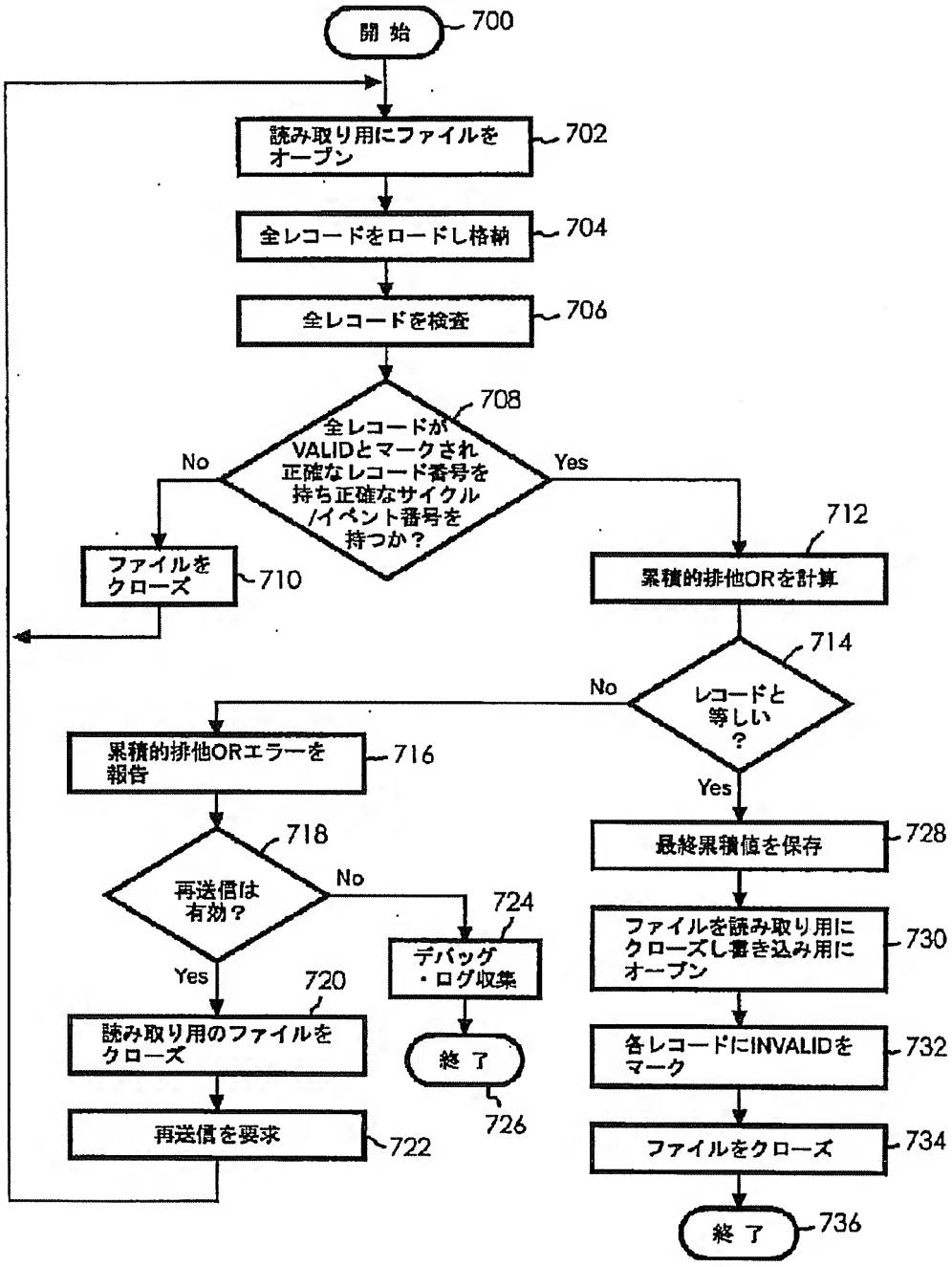
【図１０】



【図７】


開 始
500
読み取り用にファイルをオープン
502
再送信要求がないか各レコードを検査
504
ある？
506
Yes
No
再送信は有効？
520
No
Yes
ファイルをクローズ
522
ファイルをクローズし再送信を実行
526
終 了
524
各レコードにINVALIDがマーク？
508
No
Yes
ファイルをクローズ
510
ファイルを読み取り用にクローズし書き込み用にオープン
512
各レコードに書き込み
514
最終レコードに書き込み
516
ファイルをクローズ
518
終 了
528

【図９】



フロントページの続き

(72)発明者 ウィリアム・ケイ・メローズ
アメリカ合衆国 ニューヨーク州 12514、クリントン コーナーズ、トレッスル レーン 67

(72)発明者 マーヴィン・ジェイ・リッチ
アメリカ合衆国 ニューヨーク州 12603、ポキプシー、バード レーン 7

Ｆターム(参考) 5B046 AA08 BA03 CA06 JA05